

Game Semantics for Access Control

Samson Abramsky and Radha Jagadeesan

January 22, 2014

Abstract

We introduce a semantic approach to the study of logics for access control and dependency analysis, based on Game Semantics. We use a variant of AJM games with explicit justification (but without pointers). Based on this, we give a simple and intuitive model of the information flow constraints underlying access control. This is used to give strikingly simple proofs of *non-interference theorems* in robust, semantic versions.

1 Introduction

In recent years, there has been a significant development of constructive logics and type theories for access control [1, 19, 20]. The core structure of these logics has turned out to coincide in large part with a calculus previously developed as a basis for various forms of dependency analysis in programming languages [3]. This structure can be described quite succinctly as follows. We take a standard type theory as a basic setting. This may be the simply-typed or polymorphic λ -calculus [1], or some form of linear type theory [19]. Such type theories correspond to systems of logic under the Curry-Howard correspondence. We then extend the type theory with a family of monads, indexed by the elements of a “security lattice” \mathcal{L} [1]. This lattice can be interpreted in various ways. The basic reading is to think of the elements of the lattice as indicating *security levels*. We shall follow the convention, as in [1], that a higher level (more trusted) is *lower down* in the lattice ordering.

The reading which is often followed is to think of an underlying partially ordered set of “principals”, with the lattice elements corresponding to sets of principals. This leads to the reading of $T_\ell A$, where T_ℓ is the monad indexed at the security level ℓ , as “ ℓ says A ”. The monads are type-theoretic counterparts of logical modalities [26]; their use is well established both in logical type theories and in programming languages [33, 18, 12]. We illustrate their use in the specification of access control with an example drawn from Garg and Abadi [GA08].

Example 1.1 ([GA08]) *Let there be two principals, Bob (a user) and admin (standing for administration). Let $dfile$ stand for the proposition that a certain file should be deleted. Consider the collection of assertions:*

1. *(admin says $dfile$) $\Rightarrow dfile$*
2. *admin says ((Bob says $dfile$) $\Rightarrow dfile$)*
3. *Bob says $dfile$*

Using the unit of the monad with (iii) yields (admin says (Bob says dfile)). Using modal consequence with (ii) yields:

- (admin says (Bob says dfile)) \Rightarrow (admin says dfile)

dfile now follows using *modus ponens*.

The main results which are obtained in this setting, as a basis for access control or dependency analysis, are *non-interference theorems*¹, which guarantee the *absence* of information flows or logical dependencies which would contradict the constraints expressed by the security lattice \mathcal{L} . A typical example of a non-interference theorem could be expressed informally as follows:

No proof of a formula of the form “P says ϕ ” can make any essential use of formulas of the form “Q says ψ ” unless Q is at the same or higher security level as P. In other words, we cannot rely on a lower standard of “evidence” or authorization in passing to a higher level.

In the flow analysis context, it is natural to think of the constraints as ensuring that information does not flow from “higher” to “lower” variables [3]. We would then use the same definitions, and obtain the same results — but with the opposite reading of the security lattice!

Thus far, access control logics have predominantly been studied using proof-theoretic methods. Our aim in the present paper is to initiate a semantic approach based on *Game Semantics*.

Game Semantics has been developed over the past 15 years as an approach to the semantics of both programming languages and logical type theories [9, 10, 25, 11, 27, 8, 22, 7, 38]. It has yielded numerous full abstraction and full completeness results, in many cases the only such results which have been obtained. There has also been an extensive development of algorithmic methods, with applications to verification [21, 6, 34, 35, 29].

Our aim in the present paper is to show that Game Semantics provides an intuitive and illuminating account of access control, and moreover leads to strikingly simple and robust proofs of interference-freedom.

General Advantages of the Semantic Approach Proof-theoretic approaches to negative results such as non-interference properties necessarily proceed by induction over the proof system at hand. This embodies a “closed world assumption” that the universe is inductively generated by the syntax of the system, which means that each new system requires a new proof. A semantic approach, which is carried out in a semantic framework capable of providing models for a wide range of systems, and which supplies *positive reasons* — structural properties and invariants of the semantic universe — for the negative results, can be more general and more robust. We shall give an illustrative example of how semantic non-interference results can be used to obtain results about syntactic calculi in Section 5.

¹The term ‘interference’ is used in a number of senses in the security literature. Our usage follows that in [1, 19, 20].

Specific Features of the Games Model A number of features of the version of Game semantics which we shall use in this paper are interesting in their own right, and will be developed further in future work.

- We shall introduce a novel version of AJM games [10] which has a notion of *justifier*, which will be used in the modelling of the access control constraints. This notion plays an important rôle in HO-games [25], the other main variant of game semantics, but it assumes a much simpler form in the present setting (in particular, no pointers are needed). This is a first step in the development of a common framework which combines the best features of both styles of game semantics.
- We also achieve a considerable simplification of the treatment of strategies in AJM games. In particular, we eliminate the need for an “intensional equivalence” on strategies [10].

The further contents of the paper are as follows. In Section 2, we develop our variant of AJM games. In Section 3, we describe the games model for access control. We give a semantic treatment of non-interference theorems in Section 4. The relation to syntax is discussed briefly in Section 5. Finally, in Section 6 we conclude with a discussion of directions for future work.

2 Justified AJM Games

In this section, we shall introduce a minor variant of AJM games [10] which will provide a basis for our semantics of access control, while yielding a model of Intuitionistic Linear Logic and related languages isomorphic to that given by the usual AJM games.

We wish to refine AJM games by introducing a notion of *justifying move*. A clear intuition for this notion can be given in terms of procedural control-flow. A call of procedure P will have as its justifier the currently active call of the procedure in which P was (statically) declared. Thus the justifier corresponds to the link in the “static chain” in compiler terminology for ALGOL-like languages [13]. A top-level procedure call will have no justifier — it will be an “initial action”. Finally, a procedure return will have the corresponding call as its justifier.

In AJM games, moves are classified as *questions* or *answers*, and equating procedure calls with questions and returns with answers, we get the appropriate notion of justifier for games.

The notion of justifier plays a central rôle in Hyland-Ong (HO) games [25]. In that context, the identification of the justifier of a move in a given play involves an additional structure of “justification pointers”, which are a considerable complication. The need for this additional structure arises for two reasons:

- Firstly, the treatment of *copying* in HO games allows multiple occurrences of the same move in a given play. This means that extra structure is required to identify the “threads” corresponding to the different copies. By contrast, plays in AJM games are naturally *linear*, *i.e.* moves only occur once, with the threads for different copies indicated explicitly.
- The other source of the need for explicit indication of justifiers in plays is that the justification or enabling relation is in general not functional in HO games; a given move may have several possible justifiers, and we must indicate explicitly which one applies. In fact, in the original version of HO games [25], justification *was* functional;

the relaxation to more general enabling relations was introduced later for convenience [32], given that in the HO format, justification pointers were going to be used anyway.

It turns out that unique justifiers can be defined straightforwardly in AJM games; the only change which is required is a minor one to the definition of linear implication. This follows the device used in [25] to preserve the functionality of justification.

Given that we have both linearity of plays and unique justifiers, we get a very simple, purely “static” notion of justification which is determined by the game, and requires no additional information at the level of plays. The resulting notion of AJM games is equivalent to the standard one as a model of ILL, but carries the additional structure needed to support our semantics for access control.

We shall now proceed to describe the category of justified AJM games. Since a detailed account of the standard AJM category can be found in [10] and the differences are quite minor, we shall only provide a brief outline, emphasizing the points where something new happens.

2.1 The Games

A game is a structure $A = (M_A, \lambda_A, j_A, P_A, \approx_A)$, where

- M_A is the set of moves.
- $\lambda_A : M_A \rightarrow \{P, O\} \times \{Q, A\}$ is the labelling function.

The labelling function indicates if a move is by Player (P) or Opponent (O), and if a move is a question (Q) or an answer (A). The idea is that questions correspond to requests for data or procedure calls, while answers correspond to data (*e.g.* integer or boolean values) or procedure returns. In a higher-order context, where arguments may be functions which may themselves be applied to arguments, all four combinations of Player/Opponent with Question/Answer are possible. Note that λ_A can be decomposed into two functions $\lambda_A^{PO} : M_A \rightarrow \{P, O\}$ and $\lambda_A^{QA} : M_A \rightarrow \{Q, A\}$.

We write

$$\begin{aligned} \{P, O\} \times \{Q, A\} &= \{PQ, PA, OQ, OA\} \\ M_A^P &= \lambda_A^{-1}(\{P\} \times \{Q, A\}), \quad M_A^O = \lambda_A^{-1}(\{O\} \times \{Q, A\}) \\ M_A^Q &= \lambda_A^{-1}(\{P, O\} \times \{Q\}), \quad M_A^A = \lambda_A^{-1}(\{P, O\} \times \{A\}) \end{aligned}$$

and define

$$\begin{aligned} \overline{P} &= O, \quad \overline{O} = P, \\ \overline{\lambda_A^{PO}(a)} &= \overline{\lambda_A^{PO}(a)}, \quad \overline{\lambda_A} = \langle \overline{\lambda_A^{PO}}, \lambda_A^{QA} \rangle. \end{aligned}$$

- The justification function $j_A : M_A \rightharpoonup M_A$ is a partial function on moves satisfying the following conditions:
 - For each move m , for some $k > 0$, $j_A^k(m)$ is undefined, so that the forest of justifiers is well-founded. A move m such that $j_A(m)$ is undefined is called *initial*; we write Init_A for the set of initial moves of A .
 - P -moves must be justified by O -moves, and vice versa; answers must be justified by questions.

- Let M_A^\otimes be the set of all finite sequences s of moves satisfying the following conditions:
 - (p1) **Opponent starts** If s is non-empty, it starts with an O-move.
 - (p2) **Alternation** Moves in s alternate between O and P.
 - (p3) **Linearity** Any move occurs at most once in s .
 - (p4) **Well-bracketing** Write each answer a as $)_a$ and the corresponding question $q = j_A(a)$ as $(_a$. Define the set W of *well-bracketed strings* over A inductively as follows:
 $\varepsilon \in W$; $u \in W \Rightarrow ({}_a u)_a \in W$; $u, v \in W \Rightarrow uv \in W$. Then we require that s is a prefix of a string in W .
 - (p5) **Justification** If m occurs in s , $s = s_1 m s_2$, then the justifier $j_A(m)$ must occur in s_1 .

Then P_A , the set of *positions* of the game, is a non-empty prefix-closed subset of M_A^\otimes .

The conditions (p1)–(p5) are global rules applying to all games.

- \approx_A is an equivalence relation on P_A satisfying

$$\begin{aligned}
 \text{(e1)} \quad & s \approx_A t \implies \lambda_A^*(s) = \lambda_A^*(t) \\
 \text{(e2)} \quad & s \approx_A t, s' \sqsubseteq s, t' \sqsubseteq t, |s'| = |t'| \implies s' \approx_A t' \\
 \text{(e3)} \quad & s \approx_A t, sa \in P_A \implies \exists b. sa \approx_A tb.
 \end{aligned}$$

Here λ_A^* is the extension of λ_A to sequences; while \sqsubseteq is the prefix ordering. Note in particular that (e1) implies that if $s \approx_A t$, then $|s| = |t|$.

If we compare this definition to that of standard AJM games, the new component is the justification function j_A . This allows a simpler statement of the well-bracketing condition. The new conditions on plays are Linearity and Justification. These hold automatically for the interpretation of ILL types in AJM games as given in [10].

2.2 Constructions

We now describe the constructions on justified AJM games corresponding to the ILL connectives. These are all defined exactly as for the standard AJM games in [10], with justification carried along as a passenger and defined in the obvious componentwise fashion, with the sole exception of linear implication.

Times We define the tensor product $A \otimes B$ as follows.

- $M_{A \otimes B} = M_A + M_B$, the disjoint union of the two move sets.
- $\lambda_{A \otimes B} = [\lambda_A, \lambda_B]$, the source tupling.
- $j_{A \otimes B} = j_A + j_B$.
- $P_{A \otimes B} = \{s \in M_{A \otimes B}^\otimes \mid s \upharpoonright A \in P_A \wedge s \upharpoonright B \in P_B\}$.
- $s \approx_{A \otimes B} t$ iff $s \upharpoonright A \approx_A t \upharpoonright A \wedge s \upharpoonright B \approx_B t \upharpoonright B \wedge \text{out}^*(s) = \text{out}^*(t)$.

Here $\text{out} : \Sigma_{i \in I} X_i \rightarrow I :: (x \in X_i) \mapsto i$ maps an element of a disjoint union to the index of its summand. Concretely in this case, $\text{out}(m)$ is 1 if $m \in M_A$, and 2 if $m \in M_B$. Note that there is no need to formulate a ‘stack condition’ explicitly as in [10], since this is implied by the component-wise definition of the justification function.

Tensor Unit The tensor unit is given by

$$I = (\emptyset, \emptyset, \emptyset, \{\epsilon\}, \{(\epsilon, \epsilon)\}).$$

Additive Conjunction The game $A \& B$ is defined as follows.

$$\begin{aligned} M_{A \& B} &= M_A + M_B \\ \lambda_{A \& B} &= [\lambda_A, \lambda_B] \\ j_{A \otimes B} &= j_A + j_B \\ P_{A \& B} &= P_A + P_B \\ \approx_{A \& B} &= \approx_A + \approx_B. \end{aligned}$$

Bang The game $!A$ is defined as the “infinite symmetric tensor power” of A . The symmetry is built in via the equivalence relation on positions.

- $M_{!A} = \omega \times M_A = \sum_{i \in \omega} M_A$, the disjoint union of countably many copies of the moves of A . So, moves of $!A$ have the form (i, m) , where i is a natural number, called the index, and m is a move of A .
- Labelling is by source tupling:

$$\lambda_{!A}(i, a) = \lambda_A(a).$$

- Justification is componentwise: $j_{!A}(i, m) = (i, j_A(m))$.
- We write $s \upharpoonright i$ to indicate the restriction to moves with index i .

$$P_{!A} = \{s \in M_{!A}^{\otimes} \mid (\forall i \in \omega) s \upharpoonright i \in P_A\}.$$

- Let $S(\omega)$ be the set of permutations on ω .

$$s \approx_{!A} t \iff (\exists \pi \in S(\omega)) [(\forall i \in \omega. s \upharpoonright i \approx_A t \upharpoonright \pi(i)) \wedge (\pi \circ \mathbf{fst})^*(s) = \mathbf{fst}^*(t)].$$

Linear Implication The only subtlety arises in this case. The justifier of an initial move in A played within $A \multimap B$ should be an initial move in B ; but which one? To render this unambiguous, we make a disjoint copy of the moves in A for each initial move in B . A similar device is used in [25]. We write $\sum_{b \in \mathbf{Init}_B} M_A$ for this disjoint union of copies of M_A , which is equivalently defined as $\mathbf{Init}_B \times M_A$.

- $M_{A \multimap B} = (\sum_{b \in \mathbf{Init}_B} M_A) + M_B$.
- $\lambda_{A \multimap B} = [[\overline{\lambda_A} \mid b \in \mathbf{Init}_B], \lambda_B]$.
- We define justification by cases. We write m_b , for $m \in M_A$ and $b \in \mathbf{Init}_B$, for the b -th copy of m .

$$\begin{aligned} j_{A \multimap B}(m_b) &= \begin{cases} b, & m \in \mathbf{Init}_A \\ (j_A(m))_b, & m \notin \mathbf{Init}_A \end{cases} \\ j_{A \multimap B}(m) &= j_B(m), \quad m \in M_B. \end{aligned}$$

- We write $s \upharpoonright A$ to indicate the restriction to moves in $\Sigma_{b \in \text{Init}_B} M_A$, replacing each m_b by m .

$$P_{A \multimap B} = \{s \in M_{A \multimap B}^\otimes \mid s \upharpoonright A \in P_A \wedge s \upharpoonright B \in P_B\}$$

Note that Linearity for A implies that *only one copy* m_b of each $m \in M_A$ can occur in any play $s \in P_{A \multimap B}$.

- $s \approx_{A \multimap B} t$ iff $(\forall b \in \text{Init}_B) s \upharpoonright A, b \approx_A t \upharpoonright A, b \wedge s \upharpoonright B \approx_B t \upharpoonright B \wedge \text{out}^*(s) = \text{out}^*(t)$.

Note that, by **(p1)**, the first move in any position in $P_{A \multimap B}$ must be in B .

Basic Types Given a set X , we define the *flat game* X^\flat over X as follows:

- $M_{X^\flat} = \{q_0\} + X$
- $\lambda_{X^\flat}(q_0) = OQ$, $\lambda_{X^\flat}(x) = PA$ for $x \in X$.
- $j_{X^\flat}(q_0)$ is undefined (so q_0 is initial); $j_{X^\flat}(x) = q_0$.
- Plays in X^\flat are prefixes of sequences q_0x , $x \in X$.
- The equivalence \approx_{X^\flat} is the identity relation.

For example, we obtain a game **Nat**[♭] for the natural numbers.

2.3 Strategies

Our aim is to present a reformulation of strategies for AJM games, which is equivalent to the standard account in [10], but offers several advantages:

- A major drawback of AJM games is that strategies must be quotiented by an equivalence to obtain a category with the required structure. This is workable, but lacks elegance and impedes intuition. *This problem is completely eliminated here*: we present a notion of strategy which is ‘on the nose’, without any quotient.
- At the same time, the existing notions are related to the new approach, and the standard methods of defining AJM strategies can still be used.
- Although we shall not elaborate on this here, the order-enriched structure of strategies is vastly simplified, since the ordering is now simply subset inclusion.

The idea of using strategies saturated under the equivalence relation on plays can be found in [14]; but that paper concerned a ‘relaxed’ model, which could be used for Classical Linear Logic, and did not establish any relationship with the standard AJM notions.

Definition 2.1 *A strategy on a game A is a non-empty set $\sigma \subseteq P_A^{\text{even}}$ of even-length plays satisfying the following conditions:*

Causal Consistency $sab \in \sigma \implies s \in \sigma$

Representation Independence $s \in \sigma \wedge s \approx_A t \implies t \in \sigma$

Determinacy $sab, ta'b' \in \sigma \wedge sa \approx_A ta' \implies sab \approx_A ta'b'$.

To relate this to the usual notion of AJM strategies, we introduce the notion of *skeleton*.

Definition 2.2 A skeleton of a strategy σ is a non-empty causally consistent subset $\phi \subseteq \sigma$ which satisfies the following condition:

Uniformization $\forall sab \in \sigma. s \in \phi \implies \exists !b'. sab' \in \phi$.

Note that the play sab' whose existence is asserted by Uniformization satisfies: $sab \approx_A sab'$. This follows immediately from $\phi \subseteq \sigma$ and Determinacy.

Proposition 2.3 Let ϕ be a skeleton of a strategy σ . Then ϕ satisfies the following properties:

- **Functional Determinacy:** $sab, sac \in \sigma \implies b = c$
- **Functional Representation Independence:**

$$sab \in \phi \wedge t \in \phi \wedge sa \approx_A ta' \implies \exists !b'. ta'b' \in \phi \wedge sab \approx_A ta'b'.$$

Proof

- Functional Determinacy. If $sab, sac \in \phi$, then $sab \in \sigma$ and $s \in \phi$. By Uniformization, the b such that sab is in ϕ is unique, so $b = c$.
- Functional Representation Independence. Suppose that $sab \in \phi, t \in \phi, sa \approx_A ta'$. Then $sab \in \sigma$, and by **(e3)**, for some b'' , $sab \approx_A ta'b''$. By Representation Independence, $ta'b'' \in \sigma$. By Uniformization, for some unique b' , $ta'b' \in \phi$, and by the remark after the definition of skeleton, $sab \approx_A ta'b'$.

Hence we conclude. \square

Definition 2.4 We shall define a skeleton more generally — independently of any strategy — to be a non-empty, causally consistent set of even-length plays, satisfying Functional Determinacy and Functional Representation Independence.

Note that a skeleton ϕ is exactly the usual notion of AJM strategy such that $\phi \sqsubseteq \phi$ [10].

Given a skeleton ϕ , we define $\phi^\bullet = \{t \mid \exists s \in \phi. s \approx_A t\}$.

Proposition 2.5 If ϕ is a skeleton, then ϕ^\bullet is a strategy, and ϕ is a skeleton of ϕ^\bullet .

Proof We verify the conditions for ϕ^\bullet to be a strategy.

- Causal Consistency. If $ta'b' \in \phi^\bullet$, then for some $sab \in \phi$, $sab \approx_A ta'b'$. Since ϕ is causally consistent, $s \in \phi$, and $s \approx_A t$, hence $t \in \phi^\bullet$.
- Representation Independence. If $t \approx_A s \in \phi$ and $u \approx_a t$, then $u \approx_A s$, and hence $u \in \phi^\bullet$.
- Determinacy. Suppose $sab, ta'b' \in \phi^\bullet$. This means that $sab \approx_A s_1a_1b_1 \in \phi$, and $ta'b' \approx_A s_2a_2b_2 \in \phi$. Now $sa \approx_A ta'$ implies that $s_1a_1 \approx_A s_2a_2$. By Functional Determinacy and Functional Representation Independence, $s_1a_1b_1 \approx_A s_2a_2b_2$. Hence $sab \approx_A ta'b'$, as required.

Now we verify that ϕ is a skeleton of ϕ^\bullet , i.e. that Uniformization holds. Suppose that $sab \in \phi^\bullet$ and $s \in \phi$. By Functional Representation Independence, there is a unique b' such that $sab' \in \phi$ and $sab \approx_A sab'$. By Functional Determinacy, this is the unique b' such that $sab' \in \phi$. \square

Proposition 2.6 *If ϕ is a skeleton of σ , then $\phi^\bullet = \sigma$.*

Proof Certainly $\phi^\bullet \subseteq \sigma$ by Representation Independence. We prove the converse by induction on the length of $s \in \sigma$. The basis case for ε is immediate. For $sab \in \sigma$, by induction hypothesis $s \in \phi^\bullet$. Hence for some $s' \in \phi$, $s \approx_A s'$. We can find a', b' such that $sab \approx_A s'a'b'$. By Representation Independence, $s'a'b' \in \sigma$. By Uniformization, for some b'' , $s'a'b'' \in \phi$, where $s'a'b' \approx_A s'a'b''$. Hence $sab \in \phi^\bullet$. \square

Corollary 2.7 *ϕ is a skeleton of σ if and only if ϕ is a skeleton, and $\phi^\bullet = \sigma$.*

Proposition 2.8 *Every strategy σ has a skeleton ϕ .*

Proof We define a family of sets of plays ϕ_k by induction on k . $\phi_0 = \{\varepsilon\}$. To define ϕ_{k+1} , for each $s \in \phi_k$ and $a \in M_A^O$, consider the set $X_{s,a}$ of all plays in σ of the form sab for some b . Note that $(s, a) \neq (s', a')$ implies that $X_{s,a} \cap X_{s',a'} = \emptyset$. Let C be the family of all non-empty $X_{s,a}$. Then ϕ_{k+1} is a choice set for C which selects exactly one element of each member of C . Finally, define $\phi = \bigcup_{k \in \omega} \phi_k$. It is immediate from the construction that ϕ is a skeleton for σ . \square

We now recall the definition of the preorder on skeletons (standard AJM strategies) from [10]:

$$\phi \sqsubseteq \psi \equiv sab \in \phi, s' \in \psi, sa \approx s'a' \implies \exists b'. [s'a'b' \in \psi \wedge sab \approx s'a'b'].$$

Proposition 2.9 *For skeletons ϕ, ψ on A : $\phi \sqsubseteq \psi$ iff $\phi^\bullet \subseteq \psi^\bullet$.*

Proof Suppose firstly that $\phi^\bullet \subseteq \psi^\bullet$, and that $sab \in \phi$, $t \in \psi$, $sa \approx_A ta'$. Then $sab \in \psi^\bullet$, so there is some $s_1a_1b_1 \in \psi$ with $s_1a_1b_1 \approx_A sab$. Then $s_1a_1 \approx_A ta'$, so by Functional Representation Independence, there exists a unique b' such that $ta'b' \in \psi$, and $ta'b' \approx_A s_1a_1b_1 \approx_A sab$. Thus $\phi \sqsubseteq \psi$, as required.

For the converse, we assume that $\phi \sqsubseteq \psi$. It is sufficient to prove that $\phi \subseteq \psi^\bullet$, which we do by induction on the length of plays in ϕ . The base case is immediate. Now suppose that $sab \in \phi$. By induction hypothesis, $s \in \psi^\bullet$. Then for some $s' \in \psi$, $s \approx_A s'$. For some a' , $s'a' \approx_A sa$. Since $\phi \sqsubseteq \psi$, there exists b' such that $s'a'b' \in \psi \wedge sab \approx s'a'b'$. But then $sab \in \psi^\bullet$, as required. \square

We can now obtain a rather clear picture of the relationship between partial equivalence classes of strategies as in [10], and strategies and their skeletons in the present formulation.

Proposition 2.10 *For any strategy σ :*

1. *Any two skeletons of σ are equivalent.*
2. $\sigma = \bigcup \{\phi \mid \phi \text{ is a skeleton on } \sigma\}.$
3. *For any skeleton ϕ of σ : $\sigma = \bigcup \{\psi \mid \psi \approx \phi\}.$*

Proof 1. If ϕ, ψ are skeletons of σ , by Proposition 2.6 $\phi^\bullet = \psi^\bullet$, hence by Proposition 2.9, $\phi \sqsubseteq \psi$ and $\psi \sqsubseteq \phi$.

2. The right-to-left inclusion is clear. For the converse, given any $s \in \sigma$, we can guide the choices made in the construction of ϕ in the proof of Proposition 2.8 to ensure that $s \in \phi$.

3. The right-to-left inclusion follows from Proposition 2.9. The converse follows from part 2, Corollary 2.7, and Proposition 2.9. \square

Finally, we show how *history-free strategies*, which play an important rôle in AJM game semantics, fit into the new scheme. We define a strategy to be *history-free* if it has a skeleton ϕ satisfying the following additional conditions:

- $sab, tac \in \phi \implies b = c$
- $sab, t \in \phi, ta \in P_A \implies tab \in \phi$.

As in [10], a skeleton is history-free if and only if it is generated by a function on moves.

Constructions on strategies In the light of these results, we can define a strategy σ by defining a skeleton ϕ and then taking $\sigma = \phi^\bullet$. In particular, this is the evident method for defining history-free strategies. Thus all the constructions of particular strategies carry over directly from [10].

What of operations on strategies, such as composition, tensor product etc.? We can define an operation O on strategies via an operation O_{sk} on skeletons as follows. Given strategies σ, τ , we take skeletons ϕ of σ and ψ of τ , and define

$$O(\sigma, \tau) = O_{\text{sk}}(\phi, \psi)^\bullet.$$

Of course, this definition should be independent of the choice of skeletons.

Proposition 2.11 *An operation O ‘defined’ in terms of O_{sk} as above is well-defined and monotone with respect to subset inclusion if and only if O_{sk} is monotone with respect to \sqsubseteq .*

Proof Suppose that O_{sk} is monotone with respect to \sqsubseteq and that we are given $\sigma \subseteq \sigma'$, $\tau \subseteq \tau'$, and skeletons ϕ of σ , ϕ' of σ' , ψ of τ , ψ' of τ' . By Proposition 2.9, $\phi \sqsubseteq \phi'$ and $\psi \sqsubseteq \psi'$. Then $O_{\text{sk}}(\phi, \psi) \sqsubseteq O_{\text{sk}}(\phi', \psi')$, and so, using Proposition 2.9 again,

$$O(\sigma, \tau) = O_{\text{sk}}(\phi, \psi)^\bullet \subseteq O_{\text{sk}}(\phi', \psi')^\bullet = O(\sigma', \tau').$$

Note that, taking $\sigma = \sigma'$ and $\tau = \tau'$, this also shows that O is well-defined.

Conversely, suppose that O is well-defined and monotone, and consider $\phi \sqsubseteq \phi'$, $\psi \sqsubseteq \psi'$. Let $\sigma = \phi^\bullet$, $\sigma' = \phi'^\bullet$, $\tau = \psi^\bullet$, $\tau' = \psi'^\bullet$. By Proposition 2.9, $\sigma \subseteq \sigma'$ and $\tau \subseteq \tau'$. Then

$$O_{\text{sk}}(\phi, \psi)^\bullet = O(\sigma, \tau) \subseteq O(\sigma', \tau') = O_{\text{sk}}(\phi', \psi')^\bullet$$

so by Proposition 2.9, $O_{\text{sk}}(\phi, \psi) \sqsubseteq O_{\text{sk}}(\phi', \psi')$. □

Thus again, all the operations on strategies from [10] carry over to the new scheme.

2.4 Categories of Games

We build a category \mathcal{G} :

$$\begin{aligned} \text{Objects} & : \text{Justified AJM Games} \\ \text{Morphisms} & : \sigma : A \rightarrow B \equiv \text{strategies on } A \multimap B. \end{aligned}$$

Copy-Cat The *copy-cat strategy* [9] is defined by:

$$\text{id}_A = \{s \in P_{A \multimap A}^{\text{even}} \mid s|1 \approx_A s|2\}.$$

Composition We need a slight modification of the definition of composition as given in [9, 10], to fit the revised definition of linear implication. Suppose we are given $\sigma : A \multimap B$ and $\tau : B \multimap C$. We define:

$$\begin{aligned}\sigma \parallel \tau &= \{s \in M_{(A \multimap B) \multimap C}^* \mid s \upharpoonright A, B \in \sigma \wedge s \upharpoonright B, C \in \tau\} \\ \sigma; \tau &= \{s \upharpoonright A, C \mid s \in \sigma \parallel \tau\}.\end{aligned}$$

Note that

$$\begin{aligned}M_{(A \multimap B) \multimap C} &= (\Sigma_{c \in \text{Init}_C} (\Sigma_{b \in \text{Init}_B} M_A) + M_B) + M_C \\ &\cong \Sigma_{c \in \text{Init}_C} (\Sigma_{b \in \text{Init}_B} M_A) + \Sigma_{c \in \text{Init}_C} M_B + M_C\end{aligned}$$

so we can regard $s \upharpoonright B, C$ as a play in $B \multimap C$. Similarly, $s \upharpoonright A, B$, where we erase all tags from C , can be regarded as a play in $A \multimap B$. Finally, $s \upharpoonright A, C$, where all tags from B are erased, so that $(m_b)_c$ is replaced by m_c , can be regarded as a play in $A \multimap C$. This last transformation, erasing tags in B , corresponds to the elision of justification pointers in the definition of composition for HO games [25].

Proposition 2.12 *\mathcal{G} equipped with composition of strategies, and with the copy-cat strategies $\text{id}_A : A \multimap A$ as identities, is a category. There is also a sub-category \mathcal{G}^{hf} of history-free strategies.*

The further development of these categories as models of ILL proceeds exactly as in [10]. The main point to note is that monoidal closure still works, in exactly the same way as in [10]. Indeed, we have an isomorphism

$$(A \otimes B) \multimap C \cong A \multimap (B \multimap C)$$

induced concretely by the bijection on moves

$$\begin{aligned}M_{(A \otimes B) \multimap C} &= \Sigma_{c \in \text{Init}_C} (M_A + M_B) + M_C \\ &\cong \Sigma_{c \in \text{Init}_C} M_A + (\Sigma_{c \in \text{Init}_C} M_B + M_C) \\ &= M_{A \multimap (B \multimap C)}\end{aligned}$$

since the initial moves in $B \multimap C$ are just those in C . Since arrows are defined as strategies on the internal hom, this immediately yields the required adjunction

$$\mathcal{G}(A \otimes B, C) \cong \mathcal{G}(A, B \multimap C).$$

The monoidal structure for \otimes is witnessed similarly by copy-cat strategies induced by bijections on move sets, as in [9]. Thus we get a symmetric monoidal closed (SMC) category $(\mathcal{G}, \otimes, I, \multimap)$, with an SMC sub-category \mathcal{G}^{hf} .

Next we note that there are natural transformations

$$\epsilon_A : !A \rightarrow A, \quad \delta_A : !A \rightarrow !!A$$

and a functorial action for $!$ which endow it with the structure of a comonad. The counit ϵ plays copycat between A and one fixed index in $!A$, while the comultiplication δ uses a pairing function

$$p : \omega \times \omega \rightarrow \omega$$

to play copycat between pairs of indices in $!!A$ and indices in $!A$. The functorial action $!\sigma : !A \rightarrow !B$ simply plays σ componentwise in each index. The coding-dependence in these constructions is factored out by the equivalence \approx .

The co-Kleisli category $\mathbf{K}_!(\mathcal{G})$ for this comonad has arrows $!A \rightarrow B$, with identities given by the counits ϵ_A . Composition is defined via *promotion*: given $\sigma : !A \rightarrow B$, we define

$$\sigma^\dagger = \delta_A; !\sigma : !A \rightarrow !B.$$

The Kleisli composition of σ with $\tau : !B \rightarrow C$ is then $\sigma^\dagger; \tau : !A \rightarrow C$.

The additive conjunction is the product in the coKleisli category, while I is the terminal object. There are *exponential isomorphisms*

$$!(A \& B) \cong !A \otimes !B, \quad !I \cong I.$$

This ensures that the coKleisli category is cartesian closed: defining $A \Rightarrow B = !A \multimap B$, we have

$$\begin{aligned} \mathbf{K}_!(\mathcal{G})(A \& B, C) &= \mathcal{G}(!(A \& B), C) \\ &\cong \mathcal{G}(!A \otimes !B, C) \\ &\cong \mathcal{G}(!A, !B \multimap C) \\ &= \mathbf{K}_!(\mathcal{G})(A, B \Rightarrow C). \end{aligned}$$

Thus we have a symmetric monoidal closed category $(\mathcal{G}, \otimes, I, \multimap)$ and a cartesian closed category $(\mathbf{K}_!(\mathcal{G}), \&, I, \Rightarrow)$. There is, automatically, an adjunction between \mathcal{G} and its coKleisli category $\mathbf{K}_!(\mathcal{G})$. This adjunction is moreover monoidal by virtue of the exponential isomorphisms. This provides exactly the required structure for a model of ILL [16, 31]. Moreover, all this structure cuts down to the history-free strategies. The interpretation of the ILL type theory lives inside the history-free sub-category \mathcal{G}^{hf} .

3 The Model

We shall now show how a simple refinement of the games model leads to a semantics for access control.

We shall assume as given a security semilattice $(\mathcal{L}, \sqcup, \perp)$, where \sqcup is the binary join, and \perp the least element. The partial order on the semilattice is defined by

$$\ell \leq \ell' \equiv \ell \sqcup \ell' = \ell'.$$

We shall now form a category $\mathcal{G}_{\mathcal{L}}$, with a history-free sub-category $\mathcal{G}_{\mathcal{L}}^{\text{hf}}$, as follows. The objects of $\mathcal{G}_{\mathcal{L}}$ have the form

$$A = (M_A, \lambda_A, j_A, P_A, \approx_A, \text{lev}_A)$$

where $(M_A, \lambda_A, j_A, P_A, \approx_A)$ is a justified AJM game, and $\text{lev}_A : M_A \rightarrow \mathcal{L}$ assigns a security level to each move of the game. This additional piece of structure is carried through the type constructions in the simplest componentwise fashion:

$$\text{lev}_{A \otimes B} = \text{lev}_{A \& B} = [\text{lev}_A, \text{lev}_B], \quad \text{lev}_{!A} = [\text{lev}_A \mid i \in \omega]$$

$$\text{lev}_{A \multimap B} = [[\text{lev}_A \mid b \in \text{Init}_B], \text{lev}_B].$$

The remainder of the definition of $\mathcal{G}_{\mathcal{L}}$ goes exactly as for \mathcal{G} , with a single additional condition on plays in the definition of M_A^{\otimes} :

(p6) Levels A non-initial move m can only be played if $\text{lev}_A(m) \leq \text{lev}_A(j_A(m))$.

This constraint has a clear motivation, reflecting the basic intuition for access control: a principal can only affirm a proposition at its own level of authorization based on *assertions made at the same level or higher*. In terms of control flow (where the lattice has the opposite interpretation): a procedure can only perform an action *at its own security level or lower*.

Note that formally, this is a purely static constraint: it is used to discard certain moves (actions) at the level of the game (type), and independent of any particular play (run) or strategy (term). This is remarkably simple, yet as we shall see, it suffices to soundly model the formal properties of the type theories which have been proposed for access control.

The content of this constraint is essentially the same as that described at a more concrete level in [30].² What we have achieved here is to express this in a general, compositional form at the level of the semantic model. This allows general non-interference results to be proved, whereas the focus in [30] is on static analysis of specific programs.

3.1 Level Monads

For each AJM game A and $\ell \in \mathcal{L}$, there is a game A_ℓ in $\mathcal{G}_\mathcal{L}$ with $\text{lev}_A(m) = \ell$ for all $m \in M_A$. Note that, fixing ℓ , the assignment $A \mapsto A_\ell$ defines a full and faithful embedding of \mathcal{G} in $\mathcal{G}_\mathcal{L}$. The interesting structure of $\mathcal{G}_\mathcal{L}$ as a model for access control arises when there are moves at different levels.

We now define, for each $\ell \in \mathcal{L}$, a construction T_ℓ on games. This acts only on the level assignment, as follows:

$$\text{lev}_{T_\ell A}(m) = \text{lev}_A(m) \sqcup \ell.$$

All other components of A remain unchanged in $T_\ell A$. Note in particular that $P_{T_\ell A} = P_A$. We must check that the Level condition **(p6)** is satisfied by plays $s \in P_A$ with respect to $\text{lev}_{T_\ell A}$. This holds since s satisfies **(p6)** with respect to lev_A , and

$$\ell_1 \leq \ell_2 \Rightarrow \ell_1 \sqcup \ell \leq \ell_2 \sqcup \ell.$$

The following commutation properties of T_ℓ are immediate.

Proposition 3.1 *The following equations hold:*

$$\begin{aligned} T_\ell I &= I \\ T_\ell(A \otimes B) &= T_\ell A \otimes T_\ell B \\ T_\ell(A \multimap B) &= T_\ell A \multimap T_\ell B \\ T_\ell(A \& B) &= T_\ell A \& T_\ell B \\ T_\ell !A &= !T_\ell A \\ T_\ell(A \Rightarrow B) &= T_\ell A \Rightarrow T_\ell B \end{aligned}$$

The semilattice structure on \mathcal{L} acts on the \mathcal{L} -indexed family of monads in the evident fashion:

Proposition 3.2 *The following equations hold:*

$$\begin{aligned} T_\ell(T_{\ell'} A) &= T_{\ell \sqcup \ell'} A \\ T_\perp A &= A. \end{aligned}$$

²This connection was pointed out to us by one of the referees.

We can extend each T_ℓ with a functorial action: if $\sigma : A \rightarrow B$ then we can define $T_\ell \sigma : T_\ell A \rightarrow T_\ell B$ simply by taking $T_\ell \sigma = \sigma$. To justify this, note that

$$P_{A \multimap B} = P_{T_\ell(A \multimap B)} = P_{T_\ell A \multimap T_\ell B},$$

using Proposition 3.1. Hence σ is a well-defined strategy for $T_\ell A \multimap T_\ell B$.

Proposition 3.3 *The copy-cat strategy is well defined on $A \multimap T_\ell A$.*

Proof Consider a play of the copy-cat strategy

$$\begin{array}{ccc} A & \multimap & T_\ell A \\ \vdots & & \vdots \\ O & & m_1 \\ P & m_1 & \\ O & m_2 & \\ P & & m_2 \end{array}$$

which we write as $s = s_1 m'_1 m''_1 m'_2 m''_2$. If m_1 is initial, $\text{lev}_A(m_1) \leq \text{lev}_{T_\ell A}(m_1)$, so the Level condition holds for m''_1 . If m_1 is non-initial, by Justification m'_1 is preceded by its justifier m in $s|T_\ell A$. Since $s|T_\ell A \in P_{T_\ell A} = P_A$, $\text{lev}_A(m_1) \leq \text{lev}_A(m)$, so m'_1 satisfies the Level condition in this case as well. Finally,

$$\text{lev}_A(m_2) \leq \text{lev}_A(j_A(m_2)) \Rightarrow \text{lev}_{T_\ell A}(m_2) \leq \text{lev}_{T_\ell A}(j_A(m_2))$$

so m'_2 satisfies the Level condition. □

Thus we can define a natural transformation

$$\eta_A : A \rightarrow T_\ell A$$

where η_A is the copy-cat strategy. Furthermore, by Proposition 3.2, $T_\ell T_\ell A = T_\ell A$. Thus we obtain:

Proposition 3.4 *Each T_ℓ is an idempotent commutative monad.*

A similar argument to that of Proposition 3.3 yields the following:

Proposition 3.5 *If $\ell \leq \ell'$, then there is a natural transformation $\iota_A^{\ell, \ell'} : T_\ell A \rightarrow T_{\ell'} A$, where each component is the copy-cat strategy.*

4 Non-Interference Results

We now turn to the most important aspect of our semantics; the basis it provides for showing that certain kinds of data-access which would violate the constraints imposed by the security levels *cannot in fact be performed*.

Firstly, we prove a strong form of converse of Proposition 3.5.

Proposition 4.1 *If $\neg(\ell \leq \ell')$, then there is no natural transformation from T_ℓ to $T_{\ell'}$.*

Proof Suppose for a contradiction that there is such a natural transformation τ . Given any flat game X_\perp^b , with $\text{lev}_{X_\perp^b}(m) = \perp$ for all moves $m \in M_{X_\perp^b}$, the strategy $\tau_{X_\perp^b} : T_\ell X_\perp^b \rightarrow T_{\ell'} X_\perp^b$ can only play in $T_{\ell'} X_\perp^b$, since playing the initial move in $T_\ell X_\perp^b$ would violate the Level condition.

For readability, in the remainder of the proof we let $A = \mathbf{Nat}_\perp^b$. Now consider the naturality square

$$\begin{array}{ccc} T_\ell A & \xrightarrow{\tau_A} & T_{\ell'} A \\ T_\ell \sigma \downarrow & & \downarrow T_{\ell'} \sigma \\ T_\ell A & \xrightarrow{\tau_A} & T_{\ell'} A \end{array}$$

Since τ_A can only play in $T_{\ell'} A$, for all $\sigma, \sigma' : A \rightarrow A$ we have $T_\ell \sigma; \tau_A = T_\ell \sigma'; \tau_A$, and hence $\tau_A; T_{\ell'} \sigma = \tau_A; T_{\ell'} \sigma'$ by naturality. Recall that $T_{\ell'} \sigma = \sigma$. But we can take $\sigma = \{\varepsilon, q_0 0\}$, $\sigma' = \{\varepsilon, q_0 1\}$, and $q_0 0 \in \tau_A; T_{\ell'} \sigma \setminus \tau_A; T_{\ell'} \sigma'$, yielding the required contradiction. \square

The key step in the above argument was to show that control could not flow back from $T_{\ell'} X_\perp^b$ to the “context” $T_\ell X_\perp^b$ because its security level ℓ is not below ℓ' . We shall now extend this idea into an important general principle for the semantic analysis of access control.

4.1 The No-Flow Theorem

Consider the following situation. We have a term in context $\Gamma \vdash t : T$, and we wish to guarantee that t is not able to access some part of the context. For example, we may have $\Gamma = x : U, \Gamma'$, and we may wish to verify that t cannot access x . Rather than analyzing the particular term t , we may wish to guarantee this purely at the level of the types, in which case it is reasonable to assume that this should be determined by the types U and T , and independent of Γ' .

This can be expressed in terms of the categorical semantics as follows. Note that the denotation of such a term in context will be a morphism of the form $f : A \otimes C \rightarrow B$, where $A = \llbracket U \rrbracket$, $C = \llbracket \Gamma' \rrbracket$, $B = \llbracket T \rrbracket$.

Definition 4.2 Let \mathcal{C} be an affine category, i.e. a symmetric monoidal category in which the tensor unit I is the terminal object. We write $\top_A : A \rightarrow I$ for the unique arrow. We define $A \nrightarrow B$ if for all objects C , and $f : A \otimes C \rightarrow B$, f factors as

$$f = A \otimes C \xrightarrow{\top_A \otimes \text{id}_C} I \otimes C \xrightarrow{\cong} C \xrightarrow{g} B.$$

The idea is that no information from A can be used by f — it is “constant in A ”. Note that $\mathcal{G}_\mathcal{L}$ and $\mathcal{G}_\mathcal{L}^{\text{hf}}$ are affine, so this definition applies directly to our situation.

Firstly, we characterize this notion in $\mathcal{G}_\mathcal{L}$ and $\mathcal{G}_\mathcal{L}^{\text{hf}}$.

Lemma 4.3 In $\mathcal{G}_\mathcal{L}$ and $\mathcal{G}_\mathcal{L}^{\text{hf}}$, $A \nrightarrow B$ if and only if, for any strategy $\sigma : A \otimes C \rightarrow B$, σ does not play any move in A .

Proof This reduces to verifying that σ factors if and only if it plays no move in A . Certainly, if it factors it plays no move in A , since any such move in the composition must be preceded by one in I , which has none. Conversely, if it plays no move in A , then it is well-defined as a strategy $\sigma : C \rightarrow B$, and so it essentially factors through itself. \square

We now give a simple characterization for when this “no-flow” relation holds between games.

Given a game A , we define:

$$\begin{aligned} \text{Level}(A) &= \{\text{lev}_A(m) \mid m \in \text{Init}_A\} \\ A \triangleright B &\equiv \forall \ell \in \text{Level}(A), \ell' \in \text{Level}(B). \neg(\ell \leq \ell') \end{aligned}$$

Theorem 4.4 (No-Flow) *For any games A, B in $\mathcal{G}_{\mathcal{L}}$:*

$$A \not\triangleright B \iff A \triangleright B.$$

Proof If $A \triangleright B$, then any strategy $\sigma : A \otimes C \rightarrow B$ cannot play a move in A . The first such move would be an initial move in A , which would be justified by an initial move in B , and this would violate the Level condition since $A \triangleright B$.

Conversely, suppose there are initial moves m in A and m' in B such that $\text{lev}_A(m) \leq \text{lev}_B(m')$. Then for any C , $\sigma = \{\varepsilon, m'm\}$ is a strategy $\sigma : A \otimes C \rightarrow B$ which moves in A . \square

4.2 Computing Levels

The characterization of no-flow in terms of the levels of types means that we can obtain useful information by computing levels.

We consider a syntax of types built from basic types (to be interpreted as flat games at a stipulated level) using the connectives of ILL extended with the level monads. For any such type T , we can give a simple inductive definition of $\text{Level}(A)$ where $A = \llbracket T \rrbracket$:

$$\begin{aligned} \text{Level}(X_\ell^b) &= \{\ell\} \\ \text{Level}(I) &= \emptyset \\ \text{Level}(A \otimes B) &= \text{Level}(A) \cup \text{Level}(B) \\ \text{Level}(A \multimap B) &= \text{Level}(B) \\ \text{Level}(A \& B) &= \text{Level}(A) \cup \text{Level}(B) \\ \text{Level}(A \Rightarrow B) &= \text{Level}(B) \\ \text{Level}(!A) &= \text{Level}(A) \\ \text{Level}(T_\ell A) &= \{\ell \sqcup \ell' \mid \ell' \in \text{Level}(A)\} \end{aligned}$$

This yields a simple, computable analysis which by Theorem 4.4 can be used to guarantee access constraints of the kind described above.

4.3 Protected Types

We give a semantic account of *protected types*, which play a key rôle in the DCC type system [3].

Definition 4.5 *We say that a game A is protected at level ℓ if $\text{Level}(A) \geq \ell$, meaning that $\ell' \geq \ell$ for all $\ell' \in \text{Level}(A)$.*

This notion extends immediately to types via their denotations as games.

The following (used as an inductive definition of protection in [3, 1]) is an immediate consequence of the definition.

Lemma 4.6 *1. If $\ell \leq \ell'$, then $T_{\ell'} A$ is protected at level ℓ .*

2. If B is protected at level ℓ , so are $A \multimap B$ and $A \Rightarrow B$.
3. If A and B are protected at level ℓ , so are $A \& B$ and $A \otimes B$.
4. If A is protected at level ℓ , so is $!A$.
5. I is protected at level ℓ .

We also have the following *protected promotion* lemma, which shows the soundness of the key typing rule in DCC [3].

Lemma 4.7 *If $\sigma : !A \rightarrow T_\ell B$, $\tau : !B \rightarrow C$, and C is protected at level ℓ , then the coKleisli composition*

$$\sigma^\dagger; \tau : !A \rightarrow C$$

is well-defined.

Proof Firstly, by Proposition 3.1, $T_\ell !B = !T_\ell B$. So it suffices to show that τ is well-defined as a strategy $\tau : T_\ell !B \rightarrow C$. If we consider an initial move m in $T_\ell !B$ played by τ , we must have $\text{lev}_{!B}(m) \leq \text{lev}(j(m))$ since $\tau : !B \rightarrow C$ is well-defined. Moreover, $\ell \leq \text{lev}(j(m))$ since C is protected at ℓ . Hence $\text{lev}_{T_\ell !B}(m) \leq \text{lev}(j(m))$. \square

4.4 Stability Under Erasure

We now give a semantic version of the main result in [1] (Theorem 7.6), which shows stability of the type theory under erasure of level constraints. This is used in [1] to derive several other results relating to non-interference.

Firstly, given $\ell \in \mathcal{L}$, we define the erasure A^ℓ of a type A , which replaces every sub-expression of A of the form $T_{\ell'} B$, with $\ell' \geq \ell$, by \top . Semantically, this corresponds to erasing all moves m in the game (denoted by) A such that $\text{lev}(m) \geq \ell$, and all plays containing such moves.

Abadi's result is that, if we can derive a typed term in context $\Gamma \vdash e : A$, then we can derive a term $\Gamma^\ell \vdash e' : A^\ell$. To obtain an appropriate semantic version, we need to introduce the notion of *total* strategies. A strategy σ is total if when $s \in \sigma$, and $sa \in P_A$, then $sab \in \sigma$ for some b . This is the direct analogue of totality for functions, and will hold for the strategies denoted by terms in a logical type theory — although not in general for terms in a programming language equipped with general recursion. One can show that total strategies which are *finite* (or alternatively *winning*) in a suitable sense form a category with the appropriate structure to model intuitionistic and linear type theories [5, 24].

Theorem 4.8 *Suppose that $\sigma : A \rightarrow B$ is a total strategy. Then so is $\sigma' : A^\ell \rightarrow B^\ell$ for any $\ell \in \mathcal{L}$, where σ' is the restriction of σ to plays in $A^\ell \multimap B^\ell$.*

Proof Suppose for a contradiction that σ' is not total, and consider a witness $sab \in \sigma \setminus \sigma'$, with $sa \in P_{A^\ell \multimap B^\ell}$. Then $\text{lev}(b) \geq \ell$; but by the Level constraint, we must have $\text{lev}(j(b)) \geq \ell$, which by the Justification condition contradicts $sa \in P_{A^\ell \multimap B^\ell}$. \square

5 Semantic vs. Syntactic Non-Interference

Our primary emphasis in this paper is on a semantic approach to access control, and we have proved semantic versions of a number of non-interference results. A detailed analysis of how these relate to the results proved by syntactic and proof-theoretic means for calculi such as DCC would take us too far afield. However, we shall provide an illustrative example of how semantic non-interference results can be used to obtain results about syntactic calculi.

For a simple and paradigmatic example, we consider a core fragment of DCC, obtained by extending the simply-typed λ -calculus with the level monads. There are two typing rules associated with the monads:

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \eta_\ell e : A} \quad \frac{\Gamma \vdash e : T_\ell A \quad \Gamma, x : A \vdash e' : B}{\Gamma \vdash \mathbf{bind} \ x = e \ \mathbf{in} \ e' : B} \quad B \text{ is protected at level } \ell$$

The term rewriting rules, in addition to the usual β -reduction and η -expansion, are

$$\eta_\ell e \longrightarrow e \quad \mathbf{bind} \ x = e \ \mathbf{in} \ e' \longrightarrow e'[e/x]$$

Thus the normal forms in this term calculus will be the usual long $\beta\eta$ -normal forms of the simply typed λ -calculus. We say that a proof *uses an assumption* $x : A$ if the term corresponding to the proof contains x free in its normal form.

It follows from the results in Section 3 that our game semantics provides a sound model for this calculus. We have the following simple result.

Proposition 5.1 *Let $\Gamma, x : A \vdash t : B$ be a term in context of core DCC, where t is in long $\beta\eta$ -normal form. Then x occurs free in t if and only if the strategy it denotes moves in $\llbracket A \rrbracket$.*

Proof Given an occurrence of x as a head variable in some sub-term of t , we can construct a play with appropriate choices of O-moves, which will “activate” this variable, whose denotation plays a copy-cat strategy with the occurrence of x in the context, thus generating a move in A as required. The converse is easily shown by induction on normal forms. \square

Combining this with Lemma 4.3 and the No-Flow Theorem 4.4, we immediately obtain:

Proposition 5.2 *If $A \triangleright B$, any derivation of $\Gamma, x : A \vdash t : B$ cannot use the assumption $x : A$.*

Suitable adaptations of this argument to other type theories will yield corresponding non-interference results.

6 Further Directions

We have shown that Game Semantics provides a natural setting for the semantic analysis of access control. There are many further directions for this work:

- We have considered a semantic setting which is adequate for both intuitionistic and (intuitionistic-)linear type theories. It would also be interesting to look at access control in the context of *classical type theories* such as $\lambda\mu$ [37], particularly since it is suggested in [1, 20] that there are problems with access control logics in classical settings. There have been some studies of game semantics for classical type theories [36, 28]. It would be of considerable interest to see if our approach could carry over to the classical case.

- There are a number of other natural extensions, such as to polymorphic types.
- It would also be of interest to develop the applications to dependency analysis for programming languages. The same game semantics framework provides a common basis for this and the study of logical type theories.
- The development of algorithmic game semantics [21, 6, 34, 35], including several implemented verification tools [15, 17, 29], suggests that it may be promising to look at automated analysis based on our semantic approach.
- We have developed our semantics in the setting of AJM games, equipped with a notion of justification. One could alternatively take HO-games as the starting point, but these would also have to be used in a hybridized form, with “AJM-like” features, in order to provide models for linear type theories [32]. In fact, one would like a form of game semantics which combined the best features (and minimized the disadvantages) of the two approaches. Some of the ideas introduced in the present paper may be useful steps in this direction.

References

- [1] Martín Abadi. Access control in a core calculus of dependency. In John H. Reppy and Julia L. Lawall, editors, *ICFP*, pages 263–273. ACM, 2006.
- [2] Martín Abadi. Variations in access control logic. In Ron van der Meyden and Leendert van der Torre, editors, *DEON*, volume 5076 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2008.
- [3] Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *POPL*, pages 147–160, 1999.
- [4] Martín Abadi, Michael Burrows, Butler W. Lampson, and Gordon D. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.
- [5] S. Abramsky. Semantics of Interaction: an introduction to Game Semantics. In *Semantics and Logics of Computation*, ed. P. Dybjer and A. Pitts, Cambridge University Press 1997, 1–31.
- [6] S. Abramsky. Algorithmic game semantics: a tutorial introduction. In: *Proof and System-Reliability*, Kluwer, 2002.
- [7] S. Abramsky, D. R. Ghica, A. S. Murawski, I. D. B. Stark and C.-H. L. Ong. Nominal games and full abstraction for the nu-calculus. In *Proceedings LICS 2004*, 150–159.
- [8] S. Abramsky and K. Honda and G. McCusker. A fully abstract game semantics for general references. In *Proceedings LiCS 1998*, 334–344.
- [9] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic* **59**, 543–574, 1994.

- [10] S. Abramsky, R. Jagadeesan and P. Malacaria. Full abstraction for PCF. *Information and Computation* **163**, 409–470, 2000.
- [11] S. Abramsky and G. McCusker. Linearity, sharing and state. In: P. O’Hearn and R. D. Tennent, eds. *Algol-like languages*, pp. 317–348. Birkhauser, 1997.
- [12] Natasha Alechina, Michael Mendler, Valeria de Paiva and Eike Ritter. Categorical and Kripke Semantics for Constructive S4 Modal Logic. *CSL 2001*: 292–307.
- [13] A. Appel and M. Ginsburg. *Modern Compiler Implementation in C*. Cambridge University Press, 1998.
- [14] Patrick Baillot, Vincent Danos, Thomas Ehrhard and Laurent Regnier. Believe it or not, AJM’s Games Model is a Model of Classical Linear Logic. *LICS 1997*: 68–75.
- [15] Adam Bakewell and Dan R. Ghica. Game-based safety checking with Mage. *SAVCBS 2007*: 85–87.
- [16] Gavin M. Bierman. What is a Categorical Model of Intuitionistic Linear Logic? *TLCA 1995*: 78–93.
- [17] Aleksandar Dimovski and Ranko Lazic. Compositional software verification based on game semantics and process algebra. *STTT 9(1)*: 37–51 (2007).
- [18] Matt Fairtlough and Michael Mendler. An intuitionistic modal logic with applications to the formal verification of hardware. In Leszek Pacholski and Jerzy Tiuryn, editors, *CSL*, volume 933 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 1994.
- [GA08] Deepak Garg and Martín Abadi. A modal deconstruction of access control logics. In *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*, volume 4962 of *Lecture Notes in Computer Science*, pages 216–230, 2008.
- [19] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of authorization and knowledge. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 297–312. Springer, 2006.
- [20] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. *19th IEEE Computer Security Foundations Workshop (CSFW’06)*, 0:283–296, 2006.
- [21] D. R. Ghica and G. McCusker. Reasoning about idealized algol using regular languages. In *Proc. ICALP’00*, pp. 103–116. 2000. LNCS 1853.
- [22] D. R. Ghica and A. S. Murawski. Angelic semantics of fine-grained concurrency. In *Proc. FOSSACS’04*, pp. 211–225. 2004. LNCS 2987.
- [23] J.-Y. Girard, Linear Logic. *Theoretical Computer Science* 50(1):1-102, 1987.
- [24] J. M. E. Hyland. Game Semantics. In *Semantics and Logics of Computation*, ed. P. Dybjer and A. Pitts, Cambridge University Press 1997, 131–183.

- [25] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF. *Information and Computation* **163**, 285–408, 2000.
- [26] Satoshi Kobayashi. Monad as Modality. *Theor. Comput. Sci.* 175(1): 29–74 (1997).
- [27] J. Laird. Full abstraction for functional languages with control. Extended abstract, in *Proceedings LICS 1997*.
- [28] Olivier Laurent. Polarized games. *Ann. Pure Appl. Logic* 130(1-3): 79–123 (2004).
- [29] Axel Legay, Andrzej S. Murawski, Jol Ouaknine and James Worrell. On Automated Verification of Probabilistic Programs. *TACAS 2008*: 173–187.
- [30] Pasquale Malacaria and Chris Hankin. Non-Deterministic Games and Program Analysis: An Application to Security. In *Proceedings LICS 1999*: 443–452.
- [31] Maria Emilia Maietti, Paola Maneggia, Valeria de Paiva and Eike Ritter. Relating Categorical Semantics for Intuitionistic Linear Logic. *Applied Categorical Structures* 13(1): 1–36 (2005).
- [32] Guy McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Springer (1998).
- [33] Eugenio Moggi. Notions of Computation and Monads, *Inf. Comput.* 93(1): 55–92 (1991),
- [34] A. S. Murawski, C.-H. L. Ong, and I. Walukiewicz. Idealized Algol with ground recursion and DPDA equivalence. In *Proc. ICALP’05*, pp. 917–929. 2005. LNCS 35
- [35] Andrzej S. Murawski and Igor Walukiewicz. Third-Order Idealized Algol with Iteration Is Decidable. *FoSSaCS 2005*: 202–218.
- [36] C.-H. Luke Ong. A Semantic View of Classical Proofs: Type-Theoretic, Categorical, and Denotational Characterizations (Preliminary Extended Abstract). *LICS 1996*: 230–241.
- [37] Michel Parigot. Lambda-Mu-Calculus: An Algorithmic Interpretation of Classical Natural Deduction. *LPAR 1992*: 190–201.
- [38] Nikos Tzevelekos, Full abstraction for nominal general references, in *Proceedings LICS 2007*, pp. 399–410.